Questions and Answers

Subject: Using Risk Management Results to Organize Validation Efforts for Medical Device Software

Webinar Given: Thursday, September 18, 2008

Question:

What is the guideline on how many layer of Hazards one need to go to recognize user/patient harm?

Answer:

I'm not aware of any guidelines *per se*. 14971 discusses Harm/Hazard/Cause (or contributing factors). TIR32 goes a bit beyond that and discusses causal chains. I think it would be difficult to set goals for levels of relationships. More complex devices will naturally have more complex hierarchies of hazards and causes. Although the standards only provide terminology to support three layers of relationships, in my experience three is not enough layers for even moderately complex devices. In my opinion, debating whether something is a hazard or a cause is less useful than understanding the hierarchical relationships.

Question:

From the 14971 definitions overall probablity is calculated by mulitplying Probability of Hazardous situation and probability of harm. I have seen that for software people select Probablity of harm as 1. Is that a correct thing to do?

Answer:

That approach is actually recommended in one of the annexes of 14971, and is also discussed in the FDA's guidance the General Principles of Software Validation (GPSV). As we mentioned briefly in the webinar, the probability of a software failure is not random but systemic. We are not estimating the probability that a working piece of software will suddenly stop working, we are estimating the probability that a design defect exists in a given piece of software. Often software can run for years without problem, only to start having problems on the way it is used begins to migrate from the initial assumptions. So in this case, the probability that it will fail given those inputs and operating assumptions is indeed equal to one, even though the software may have run for years without problem.

Question:
Do you believe that it is possible to design regulated or medical device software that poses no risk whatsoever?

Answer:
In my opinion the answer is no; there is always some residual risk. The risks may be extremely low because our risk control measures may have reduced the probability or resultant severity of a risk to extremely low values, but it is the exception in which the probability or severity can be reduced to zero.

Question:
What are the benefit of using Hazard Analysis, FMEA, FMECA,
FTA, or multiple? When would you use it (provide an example)? How is it useful?

Answer:
All of the different analysis techniques are different in the way they look at the risk or hazard data. Risk analysis or hazard analysis tends to be more of a top-down approach which can be attempted before very little design or implementation data is available. The value of the top-down approach is precisely that it can be attempted early in the product development lifecycle. The risk control measures that are identified as an output of the process can be used as design inputs at an early enough phase of development that they can be implemented without major rework.

On the other hand, techniques like FMEA which look at failure modes of specific subsystems or components require detailed information about the design and implementation of the device. These kinds of failures generally cannot be anticipated in early phases of the product development lifecycle.

TIR32 discusses using a diversity of risk analysis techniques. Different techniques are appropriate at different phases of the lifecycle since risk management is not a one-time event, but is a supplementary process that takes place throughout the product development lifecycle.

Question:
should all complaints from field be considered as hazards and filter them down to different areas of issues like user errors, hazards, etc.

Answer:
I am not sure about *all* complaints, but certainly those complaints that are related to the occurrence of harm or the potential for harm should be factored back into the risk analysis. As we mentioned in the webinar, risk management does not stop when the product is released to the field. The example you give here is a good example of the ongoing nature of risk management. Data should be gathered from any source from which it is available, and analyzed in the context of the full risk management plan to determine if adequate risk control measures are in place to manage the severity and probability down to acceptable levels.

Question:
where as you are stating P= Pf +Ph harm on one slide and Ptot=Pf x Ph in another. Which formula one is correct?

Answer:
The correct slide is the one which indicated that the total probability is the product of the probability of failure and the probability of the hazard resulting in harm.

Question:
How do you define "Less Validation"? and How do you perform "More Validation" for high risk user needs?

Answer:
This is a great question, and one of my favorite topics. There are standards, such as 62304 which do specify which development and validation activities should be performed based on a risk classification of the device. This approach eliminates entire activities for lower risk devices. Personally, I am not a fan of this approach because I am not convinced there are any “spare parts” in validation that can be eliminated without a thoughtful analysis of what the risks and potential failures are and how they are best validated.

If we accept that risk management is a validation activity, then certainly just by the nature of the risk management process are activity level will be greater for a high risk device or a complex device than it would be for a simple low risk device.

I think it is difficult to prescribe validation activities generically based on a handful of risk categories. I would much prefer to let common sense be my guide for scaling up validation activities where the risks are highest in the probability of finding defects the greatest.

Consider, a piece of software which performs a calculation for a low risk clinical use based on several pieces of data input by the user, and another system which monitors cardiac activity and controls defibrillator energy in embedded device. The low risk device validation plan should focus a relatively large part of its activities on the computation algorithm, and may be less concerned about long-term stability, performance, and minor user interface inconveniences. The high risk embedded device, on the other hand, would have severe consequences if its monitoring or control algorithms failed, or if there were a memory leak that caused it to fail periodically. Clearly, more intense testing of the algorithms at all levels (system, integration, and unit) would be performed in addition to performance testing, reliability and stability testing, etc.

Question:
Can your Regression testing based solely on Risk Analysis, in other words include items related to risk?

Answer:
Using risk analysis to select regression suite of tests is an excellent idea. However, since regression testing is intended to cast a wide net to find problems in the areas of code that were not changed since the last release, selecting your regression suite selection based only on risk could leave you exposed to defects creeping in to less risky areas of the code.

My preference would be to select the regression suite based on risk as you suggested supplemented with test protocols for areas of code that have a higher probability of failure. Several of the slides in the webinar addressed attributes of software that make it a higher probability of failure.

Question:
last slide of part 1 stated that testing can be used us risk control measure... can you please clarify how? the verification of design to ensure that risk control measure would be appropriate, to demonstrate that one has sufficiently mitigated the hazard. Please clarify.

Answer:
If we consider that risk is a product of severity of harm times the probability of the risk occurring, testing can be viewed as a risk control measure that reduces the probability of failure. Testing is a weak risk control measure because it is not as quantifiable as control measures that reduce severity. As we stated in the webinar quantifying the probability of software failure is a difficult concept. We do not know the probability of failure of the software before we test. We do not know how much the probability has been reduced by testing (since testing is not perfect either). Therefore, we do not have a good quantifiable handle on what the residual risk of failure is after the testing is complete.

Having said all that, frequently there just are not any good risk control measures for hazards related to software failure. Computational algorithms are good example of this. It’s a relatively simple matter to design some range checking of algorithmic outputs for results that are totally outside the expected range of outputs. It is much more difficult to design a risk control measure for incorrect computational results that are within the expected range of normal outputs. In some cases like this, it may be that the only way to reduce the risk of an incorrect computational result is by comprehensive testing. In these situations, testing is the risk control measure.

Question:
How do you recommend managing risk with 3rd party software in the device or 3rd party software used to develop or manufacture device? How proactive need to be in monitoring bug releases in 3rd party software?

Answer:
The risk *analysis* with third party software would not differ much from early life cycle risk analysis for software that is developed in-house. Risk *management* would be significantly different. The tools you have available for validation and for managing risk with third party software are more limited than those that are available when you have full control of the development process.

The probability of failure is related to the development and validation processes the third party developer has in place. Therefore, the FDA in GPSV recommends that vendor audits may be appropriate to assess the risk of software failure. Other measures that help assess that risk might include the number of users of the software in the marketplace, the types of users of the software, the availability of online defect reporting of the software, etc.

In any event, if the third party software is being used for high risk application, a thorough understanding of your intended use of the software and the requirements for the software to meet their intended use is necessary. Any external risk control measures that can be implemented to contain the risk of use of the software would be the first preference. In the absence of external risk control measures, you may be left only with testing as a risk control measure. That testing should be focused on the high risk requirements of the software, and those areas that are perceived to be most likely to be defective.

Question:
Do you feel that some software RCMs can be effectively verified at the Unit test level, and consequently not tested during the system test level?

Answer:
My opinion is that testing at both the unit and system test level would be preferable. There are however, a large number of situations in which risk control measures are implemented in software and can only be tested at the unit test level. A good example of this is software implemented as traps or service codes for those unexpected system states that can "never happen". Many times the system states cannot be achieved at the system level for testing and can only be simulated in a unit test environment.

If the software risk control measure can be exercised and tested at the system level, that too, should be used as a mechanism for testing the effectiveness of the risk control measure. Often software risk control measures can be buried deep under many layers of code. To test only at the unit level, and extrapolate that its performance at the system level would be correct may be assuming too much software "above" the risk control measure is working correctly. It is never a good idea to assume that software is working correctly.

Question:
How to validate a real time operating system in a defibrillator if it isn´t validated?

Answer:
The answer to this is very similar to the answer to the question above about third party software. Since this is the riskiest of risky applications, one probably should use every mechanism at one's disposal to build confidence in the operating system.

At a minimum, I would consider the activities mentioned above. Additionally, each feature and function of the operating system that is used directly by the device application should be tested thoroughly. I thoroughly I mean tests that do more than simply exercise each feature. Features used should be driven to their performance limitations to understand the vulnerability of the device to those limitations. Reported defects on the vendor's website should be studied to determine whether any of those defects are applicable to the way in which the operating system is used in the device. By the way, if you do all these risk management activities, do not forget to document them.
